Available online at www.sciencedirect.com

SciVerse ScienceDirect

Physics Procedia 24 (2012) 560 – 565

Physics Procedia

2012 International Conference on Applied Physics and Industrial Engineering

The Activity Chain Safety and Liveness Specification of Composite Web Services*

Bo Chen[a],Xiaomei Huang[b]

aDept. of Computer Engineering Guangxi University of Technology,Liuzhou, China, 545006
bThe Library of Guangxi University of Technology,Liuzhou, China, 545006

Abstract

Web service composition is most impressing method for development and deployment of e-business. Description and modeling the behavior requirements of composite Web services for users and verifying composite Web service compliance to specific requirements is an important key in design of services. But most work does not address the issue of how to model the requirements that the BPEL4WS processes are supposed to satisfy. The specifications in verification works are general temporal relation based on activity or scenario in essence. Distinguish with these work, we propose a novel concept of behavior specification based on activity chain in which granularity is between activity and scenario. Chain existence mode, chain absence mode are designed to express such behavioral requirements based on activity chain that is similar with safety or liveness specification based on activity respectively. Encode them on Labeled Transition System LTS and then give them exact operation semantics. Finally, an example is illustrated.

Keywords: activity chain; behavior mode; safety and liveness spedification; composite Web service.

1.Introduction

A Composite Web services has implemented the integration of information and reuses of software in internet. However, distributed nature of Web accessible service components makes interoperability and integration especially challenging: interface violations, concurrence errors and different interaction behaviors etc. This led to many research analysis and verification issues in theoretical and practical [1]

Tarek[1],Kazhamiakin[2] and Bordeaux[3] focused on anglicizing communication behaviors of composite Web services to answer whether one peer participated in composite Web service communicates

* Supported by Guangxi Sci. & Tec. Program under grant of 0992006-13

 doi:10.1016/j.phpro.2012.02.082

wit other peers in according with specific protocol? Two peers are compatible with each other? One peer can be replaced with others? Ouyanga[4], Nakajima[5] ,Qiu[6] checked whether a composite Web service flow is correct? Whether the constraints among components are satisfied? And verify the business flow with full support for compensation and error handling. Fu[7],Can[8] ,Mongiello[9], Foste[10] and Aalst[11] checked whether the flow behavior of a composite Web service are compliant to user' specific requirements.

However, most of existing works do not address the issues of how to model the requirements that the BPEL4WS processes are supposed to meet [12]. The behavior requirements are usually two types. One is LTL or CTL like logic formula which is temporal relations based on activity [7-9]. Another is MSC like graphic specification which is based on scenario [10-11]. Some scholars attempted to find more expressive method to describe behavior requirements, For example, Pistore[12] and Rouached [13]. But in essence, their methods are based on activity or event. In case of composite Web service, behavior requirements usually deal with multi-activities of different service components. We call them behavior specifications based on activity chain which granularities are activity chain. Expressing them with general LTL formula may lead it too complex to be understood due to its small granularity. Meanwhile MSC or automaton method can not describe more detailed requirement than scenario.

In this paper, we focus on the behavior requirements based on activity chain. How to describe such requirements with concise method? What are these requirements dynamic operation semantics? We propose chain existence mode and chain absence mode to express the behavior requirements for composite Web services. They have the properties just like the liveness and safety properties based on activities. Their exact semantics are expressed in modes LTS. We can express general requirements based on activity chain in term of these two chain modes.

2.Activity chain in composite Web service and behavior requirements

Let $WS=\{ws_1,ws_2,\ldots,ws_m, ws_{orch}\}$ represents a composite Web services where ws_{orch} is the name of orchestrator and ws_i is a service component.

Difinition2.1. $O_{WS}=\{o_{WS}|o_{WS}=op[?m] \text{ or } o_{WS}= op [!m]\}$ is the set of operations in the ports of ws, where op is a name of operation.

$A_{WS}=\{a_{WS}|a_{WS}=$ receive[o]ws_{orch} or reply[o]ws_{orch} or invoke[o]$ws_{orch.}\}$is the set of basic activities of service ws.

$A_{orch}=\{a_{orch}|a_{orch}=$receive[o]ws or reply[o]ws or invoke $[o_{WS}]$ $\}$is the set of basic activities of orchestrator.

$O=\cup\{o_{WS}|ws\in WS\}$ is the set of operations in a composite Web service

$Act=\cup$ $\{a_{WS}|$ $a_{WS}\in A_{WS}, ws\in WS\}$ is the set of basic actitities of a composite Web service. We also omit the subscript of a_{WS} in cases without ambiguity.

Difinition2.2. finite activities tuple $C=<a_1, \ldots, a_n>$, where $a_i\in Act$, $1\le i\le n$, a_{i+1} immediately follows a_i in an execution of Composite Web service WS is called an activity chain.

An infinite or finite activity chain $\sigma=<\sigma_1, \ldots,\sigma_z, \ldots>,\sigma_i\in Act$, $i.\ge 1$.is a trace of an execution of a composite Web service when it just contains all the activities occurred in an execution order of composite Web service.

let $C=<a_1, \ldots, a_n>$ an activity chain and $\sigma=<\sigma_1, \ldots,\sigma_z, \ldots>$ a trace of WS. a is an activity of WS.

Definition2.3. C and σ are of chain existence relation, written C ***C-EX*** σ, if there is a finite subtrace $<\sigma_{i1}, \ldots,\sigma_{in})>$ of σ satisfies that $\sigma_{ij}=a_j,1\le j\le n$. we say C is of chain existence mode globally if C ***C-EX*** σ hold for any trace σ, written C ***C-EX*** Globally and C is named required.

Definition2.4. C and σ are of chain absence relation, written C ***C-AB*** σ, if there is no any finite subtrace $<\sigma_{i1}, \ldots,\sigma_{in})>$ of σ satisfies $\sigma_i=C$. C is of chain absence mode globally if C ***C-AB*** σ holds for any trace σ and C is named unreachable.

Chain existence mode is a concise method for users to express such a kind of requirement based on activity chain that can be regarded as an extension of liveness property based on activity. Meanwhile chain absence mode can be regarded as extension of safety property based on activity.

3.The semantics of activity chain modes

The labeled transition system (LTS) is widely used to describe the dynamic semantic of distributed concurrent system [14]. In this section, we encode the activity sequence modes presented in section 2 into LTS and give these modes the precise interpretation.

Definition3.1 An LTS is a tuple L=(S,A,→,s),where S is the set of finite states. A=αL ⊆Act is the set of finite activities. →⊆S×A_τ×S is a transition relation. A_τ=A ∪ {τ}. s is initial state and τ is internal activity that is invisible to extern.

When L executes an activity a, a∈Aτ, (s, a, s')∈→, then it may become L', L'=(S, A,→, s'). Denote it $L \xrightarrow{a} L'$, iff $s \xrightarrow{a} s'$, here, $s \xrightarrow{a} s'$ is the same mean that of (s, a, s')∈→.

Definition3.2.Let LTS L_1,L_2 are two LTS. The parallel of two LTS is the LTS L denoted as $L=L_1||L_2$. The rules of parallel operation of two LTS are listed below..

$$(1)\frac{L_1 \xrightarrow{a} L_1'}{L_1 \| L_2 \xrightarrow{a} L_1' \| L_2}, a \notin \alpha L_2 \qquad (2)\frac{L_2 \xrightarrow{a} L_2'}{L_1 \| L_2 \xrightarrow{a} L_1 \| L_2'}, a \notin \alpha L_1 \qquad (3)\frac{L_1 \xrightarrow{a} L_1', L_2 \xrightarrow{a} L_2'}{L_1 \| L_2 \xrightarrow{a} L_1' \| L_2'}, a \in \alpha L_1 \cap \alpha L_2$$

In this paper, a Web service is expressed as a LTS and a composite Web service is expressed as the parallel of finite LTSs that is $L_W==L_1||L_2||\ldots||L_k$, L_i, 1≤i≤k, presents a service component. Then the execution and its trace of a composite Web service can be defined below.

Definition3.3. Let L=<S,A,→,s_0>,A=αL, is a LTS. ρ=$s_0a_1s_1a_2\ s_2\ldots$is an infinite or finite alternating sequence of states and activity labels, where s_i∈S,i≥0,a_j∈A,j≥1, $s_{i-1} \xrightarrow{a} s_i$, i≥1. ρ is called an execution of L. σ=<a_1,$a_2\ldots$> is called the trace corresponding to ρ.

Definition3.4. Let L=<S,A,→,s_0>,A=αL, is a LTS. s∈S, a∈αL ∪ {τ}, Post(s, a)={s'| $s \xrightarrow{a} s'$} is the set of direct successor states of s related to activity a. Post(s) =$\cup_{a\in\alpha L\cup\{\tau\}}$Post(s, a) is the set of direct successor states of s

A state s is called termination state of L when Post(s) =∅. An execution of L is called finite termination iff after finite steps of execution of L, ρ=$s_0a_1s_1\ldots s_n$ and Post (s_n) =∅.:

In order to facilitate users to express the behavior requirements based on activity chain, we will give the exact meaning of every mode by interpreting its semantic through the mapping rule. The mapping rules from behavior modes to LTS have been listed in figure 3.1below. The mapping rules from behavior modes to LTS have been listed in figure 3.1below.

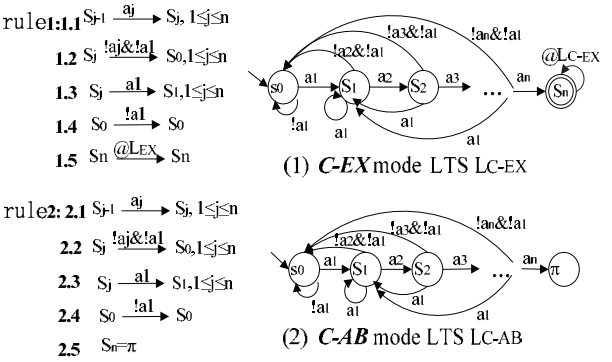


Figure 3.1 chain modes mapping rules and LTS

In general, a specific LTS presents a ceaseless reactive system which its executive path is infinite. Thus if it goes into a terminate state, that is Post(s) =Φ, s must be its deadlock state. However, a LTS presents a Web service or its chain modes in this paper and its normal execution may be finished after running infinite or finite activities. If a state s is a terminate state, it may be possible a deadlock state or possible normal finite finish state. In order to verify the compliance with chain modes for composite Web services in future work, we extend chain existence mode LTS to capture its normal finished execution. In figure 3.1(1), the LTS is extended with self-loop transition labeled with $@L_{C\text{-}EX}$ in terminate state s_n. The self-loop transition is called receive transition. $@L_{C\text{-}EX}$ is called receive activity and s_n is called receive state.

Similarly, We extend chain absence mode LTS to capture its absence character that means its specific state is not reachable. In figure 3.1(2), the LTS is extended with error state. The detailed explain can be seen definition 3.5 and definition 3.6.

In order to check the compliance to chain absence mode for a composite Web service, we extend the chain absence mode LTS.

Definition3.5 let LTS $L=<S,A,\rightarrow,s_0>$ is a LTS. $\Pi=<\{\pi\},A,\varnothing,\pi>$ is LTS that is trapping into an error state and π is an error state meaning that the Π can not go further more.

Definition3.6 .let LTS $L=<S,A,\rightarrow,s_0>$ is a activity chain absende mode LTS. $L'=<S\cup\{\pi\},A,\rightarrow',s_0>$ where $\rightarrow'$ is $\rightarrow\cup\{s\xrightarrow{a}\pi|s\in S, a\in A, \nexists s'\in S.\ s\xrightarrow{a}s'\}$. L' is called as image LTS.

In figure 3.1(2), $L_{C\text{-}AB}$ is an image LTS which is express the property based on activity chain should not to occur.

4.Example and analysis

Example: Flight and Hotel are two existed services, which provide separately flight and hotel booking service for the Client. Travel agency F_H is the composite Web service orchestrator, which provides integrated service for client. F_H is responsible for invocation operations of Flight and Hotel. Figure 4.1 describes an interactions scenario. The safety and liveness property based on activity chain are listed below

R: (1) The service must start to invoke flights filter after it has received a request of a client.

(2)After finish flights filter service, it must not provide flights sort service.

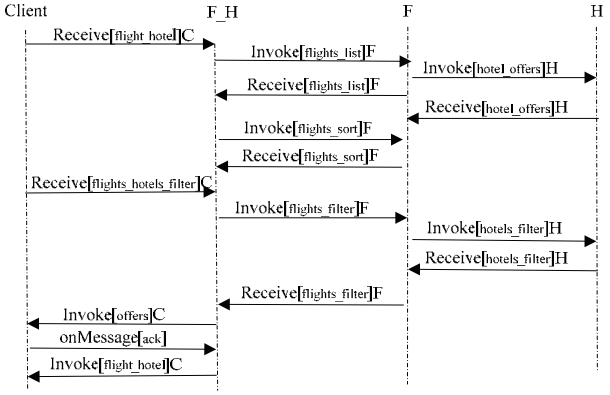


Figure 4.1 An execution scenario of F_H

The behavior requirements above are expressed in such chain modes below

(1) C_1 ***C-EX*** Globally，C_1=<receive[flights_hotels_ filter]C, invoke[flights_filter]F>

(2) C_1 ***C-AB*** Globally，C_1=< receive[flights_filter]F, invoke[flights_sort]F>

Figure 4.2 is the LTS of composite Web service F_H, chain existence mode LTS, chain absence mode LTS.

The LTS of F_H is obtained by translating BPEL process of F_H into LTS [10]. For simplicity, we omit the LTSs of two services, F and H, and their parallel. Because in example, the composite service is in orchestration synthesized. A client, F and H communicate with only orchestrator F_H. services. The activities related with behavior requirements are all in F_H service. The figure 4.2(2) and (3) are chain mode LTSs of R.(1) and (2) respectively in example.

5.Conclusion

In this paper, we propose the concept of behavior specification based on activity chain. Its granularity is between activity and scenario. Behavior requirements based on activity chain are temporal relations between activity chains. We propose two chain modes that are chain existence mode and chain absence mode to express certain properties based on activity chain. These two chain modes are regarded as extensions of safety property and liveness property based on activity in usual. Chain modes facilitate users to express such behavior requirements in concise mode. Its idea is come from the attribute patterns [15,16]. But in chain modes, the scope is not adopted obviously. The reason is that with a the scope the mode would be more complex and scope can be regarded as other modes that will be discussed in our other work. Future work may be consideration of the sufficient and necessary condition for the compliance verification to chain modes.

References

[1]M.Tarek, C. Boutrous-Saab. Verifying correctness of Webservices choreography. Proceedings of the European Conference on Web Services, 2006, pp306-318.

[2]R.Kazhamiakin, M. Pistore. A parametric communication model for the verification of BPEL4WS Compositions. Lecture Notes in Computer Science, 2005, v3670, pp318-332.

[3]L.Bordeaux, G.Salaun, D.Berardi,M. Mecella. When are two Web services compatible? Lecture Notes in Computer Science, 2005, v3324, pp15-28.

[4]C.Ouyanga,_E. Verbeekb, V.D. Aalsta. Formal semantics and analysis of control flow in WS-BPEL. Science of Computer Programming, 2007, 67(4):162–198.

[5]S.Nakajima. Model-checking behavioral specification of BPEL application. Electronic Notes in Theoretical Computer Science, 2006, 151(2):89-105.

[6]Z.Qiu, S.Wang, G.Pu, X.Zhao. Semantics of BPEL4WS-like fault and compensation handling. Lecture Notes in Computer Science, 2005, v3582, pp350-365.

[7]X.Fu, T. Bultan, J.Su. Analysis of interacting BPEL Web services. Proceedings of the 13th International Conference on World Wide Web, 2004, pp 624-630.

[8]A.B.Can, T.Bultan, X.Fu. Design for verification for asynchronously communicating Web services. Proceedings of the 14th International Conference on World Wide Web, 2005, pp750-759.

[9]M. Mongiello, D.Castelluccia. Modeling and verification of BPEL business processes. Proceedings of the 4th Workshop on Model-Based Development of Computer-Based Systems.2006, pp144-148

[10] H.Foster, S.Uchitel, J.Magee, J. Kramer. Model-based verification of Web service compositions. IEEE International Conference on Automated Software Engineering, 2003, pp152-163.

[11] V.D. Aalst. Conformance checking of service behavior. ACM Transactions on Internet Technology, 2008, 8(3):1-13.

[12] M. Pistore, M. Roveri, P.Busetta. Requirements-driven verification of Web services. Electronic Notes in Theoretical Computer Science, 2004, 105(3):95–108.

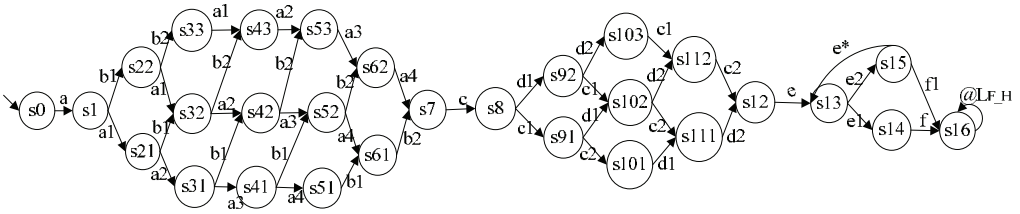
[13] M.Rouached, C. Godart. Requirements-driven verification of WSBPEL processes. IEEE International Conference on Web Services, 2007, pp354–363.

[14] D. Giannakopoulou. Model checking for concurrent software architectures. Imperial College of Science, Technology and Medicine University of London. Ph.D.thesis, 1999.

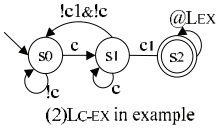
[15] M.B. Dwyer, G.S. Avrunin, J.C. Corbett. Patterns in property specifications for finite-state verification. Proceedings of the 1999 International Conference on Software Engineering, 1999, pp411-420.

[16] J. Yu, T.P. Manh, J. Han, U. Jin, Y. Han, J.W. Wang. Pattern based property specification and verification for service composition. Lecture Notes in Computer Science, 2006, v4255, pp156-168.

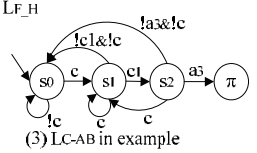
a=receive[flight_hotel]C
a1=Invoke[flights_list]F
a2=receive[flights_list]F
a3=invoke[flights_sort]F
a4=receive[flights_sort]F
b1=Invoke[hotel_offers]H
b2=receive[hotel_offers]H
c=receive[flights_hotels_filter]C
c1=invoke[flights_filter]F
c2=receive[flights_filter]F
d1=invoke[hotels-fiter]H
d2=receive[hotels_filter]H
e=invoke[offers]C
e1=onMessage[ack]C
e2=onMessage[nack]C
e*=invoke[offers_next]C
f=invoke[flight_hotel]C
f1=invoke[nack]C



(1) The composite web service F_H and LTS L_{F_H}



(2)$L_{C\text{-}EX}$ in example



(3) $L_{C\text{-}AB}$ in example

Figure 4.2 The LTSs of composite Web service F_H, chain modes and their parallel

.